Forward, Tangent Linear, and Adjoint Runge-Kutta Methods in KPP–2.2

Philipp Miehe and Adrian Sandu

Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
{pmiehe, asandu}@cs.vt.edu

Abstract. This paper presents the new stiff solvers of the new version 2.2 of the Kinetic PreProcessor (KPP). Taking a set of chemical reactions and their rate coefficients as input, KPP generates Fortran90, Fortran77, Matlab, or C code for the temporal integration of the kinetic system. Efficiency is obtained by carefully exploiting the sparsity structures of the Jacobian and of the Hessian. A set of integration methods was added to the comprehensive suite of stiff numerical integrators. Moreover, KPP is now ready do be used to generate the tangent linear model, as well as the continuous and discrete adjoint models of the chemical system to do sensitivity analysis.

1 Introduction

The application of computer modeling in atmospheric chemistry requires efficient tools for simulation and analysis of chemical reaction mechanisms. Due to the side by side existence of very stable (e.g., CH_4) and very reactive (e.g., O^{1D}) species numerically challenging software has been developed to integrate stiff ordinary differential equations (ODEs), e.g., Facsimile [6], AutoChem [19], Spack [7], Chemkin [20], Odepack [21] , and KPP [1].

Many academic, research, and industry groups in different countries are currently using KPP [8, 9, 11, 12, 10]. The Master Chemical Mechanism (MCM [22]) now provides the option for output in KPP syntax. In this paper we focus on new integrators introduced in version 2.2 of KPP. These integrators allow for high accuracy in efficient simulation of chemical kinetic systems in Fortran90 and Matlab.

The paper is organized as follows. A short overview of usage for KPP is given. The new integrators are presented in Sec. 3. Here we show the mathematical background and mention implementation details. Section 4 shows results from applying the solvers to a chemical system. The conclusion finally is presented in Sec. 5.

2 The Kinetic PreProcessor KPP

The Kinetic PreProcessor KPP [1, 2, 3] is a software tool that assists the computer simulation of chemical kinetic systems. The concentrations of a chemical

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 120–127, 2006.

system evolve in time according to the differential law of mass action kinetics. A numerical simulation requires an implementation of the differential laws and a numerical integration in time. KPP is currently being used by many academic, research, and industry groups in several countries [8, 9, 11, 12, 10]. The well-established Master Chemical Mechanism (MCM [22]) has also recently been modified to add the option of producing output in KPP syntax.

KPP translates a specification of the chemical mechanism into Fortran77, Fortran90, C, or Matlab simulation code that implements the concentration time derivative function, its Jacobian, and it Hessian, together with a suitable numerical integration scheme. Sparsity in Jacobian/Hessian is carefully exploited in order to obtain computational efficiency. Fortran90 is the programming language of choice for the vast majority of scientific applications. Matlab [25] provides a high-level programming environment for algorithm development, numerical computations, and data analysis and visualization. The Matlab code produced by KPP allows a rapid implementation and analysis of a specific chemical mechanism.

KPP incorporates a library with several widely used atmospheric chemistry mechanisms; the users can add their own chemical mechanisms to the library. KPP also includes a comprehensive suite of stiff numerical integrators. The KPP development environment is designed in a modular fashion and allows for rapid prototyping of new chemical kinetic schemes as well as new numerical integration methods.

A summary of KPP generated routines is given below:

1. *Fun*: the time derivative of concentrations;
2. *Jac, Jac_SP*: Jacobian of Fun in full or in sparse format;
3. *KppDecomp*: sparse LU decomposition for the Jacobian;
4. *KppSolve, KppSolveTR*: solve sparse system with the Jacobian matrix and its transpose;
5. *Jac_SP_Vec, JacTR_SP_Vec*: sparse Jacobian (transposed or not) times vector;
6. The stoichiometric matrix STOICM;
7. *ReactantProd*: vector of reaction rates;
8. *JacReactantProd*: the Jacobian of the above;
9. *dFun_dRcoeff*: derivatives of Fun with respect to reaction coefficients (in sparse format);
10. *dJac_dRcoeff*: derivatives of Jac with respect to reaction coefficients times user vector;
11. *Hess*: the Hessian of Fun; this 3-tensor is represented in sparse format;
12. *Hess_Vec, HessTR_Vec*: Hessian (or its transpose) times user vectors; same as the derivative of Jacobian (transposed) vector product times vector.

3 The Numerical Integrators

The new numerical integrators expand the original set of stiff solvers for ODEs for chemical kinetic systems. The KPP numerical library provides implementations of several stiff numerical solvers. Efficiency is obtained through the use

of sparse linear algebra routines generated by KPP. Several Rosenbrock methods of various orders are implemented in KPP [4]. These methods have proved to be very efficient on many applications, and especially in atmospheric chemistry. The variable order stiff extrapolation code SEULEX [14] is able to produce highly accurate solutions. The Livermore ODE solver (LSODE, LSODES [16]) implements backward differentiation formula (BDF) methods for stiff problems. The solver VODE [17] uses a different formulation of backward differentiation formulas. The BDF-based direct-decoupled sensitivity integrator ODESSA [15] has been modified to use the KPP sparse linear algebra routines.

In this paper we focus on the new implementations of stiff Runge-Kutta methods which have been added to the KPP library. A general s-stage Runge-Kutta method is defined as [13]

$$\begin{aligned} y^{n+1} &= y^n + h \sum_{i=1}^{s} b_i k_i \ , \quad T_i = t^n + c_i h \ , \quad Y_i = y^n + h \sum_{j=1}^{s} a_{ij} k_j \ , \\ k_i &= f\left(T_i, Y_i \right) \ , \end{aligned} \tag{1}$$

where the coefficients a_{ij}, b_i and c_i are prescribed for the desired accuracy and stability properties. The stage derivative values k_i are defined implicitly, and require solving a (set of) nonlinear system(s). Singly diagonally-implicit Runge-Kutta (SDIRK) methods [13] are defined by (1) with the coefficients $a_{ii} = \gamma$ and $a_{i,j} = 0$ for all i and $j > i$.

Two families of solvers have been added to the KPP library, with multiples methods within each family. These methods are discussed next.

Fully implicit Runge-Kutta methods. The three stage Radau-2A is a Runge-Kutta method of order 5 based on Radau-IIA quadrature and stiffly accurate. While Radau-2A is relatively expensive (when compared to the Rosenbrock methods), it is more robust and is useful to obtain accurate reference solutions. The discrete adjoint of Radau-2A methods are the Radau-1A methods with the same number of stages; therefore the discrete adjoints of Radau-2A methods are not stiffly accurate. The three stage Lobatto-3C method of order 4 is another stiffly accurate method, whose discrete adjoint is the Lobatto-3C method

Table 1. The coefficients of Radau-2A method (left) of order 5 and of the Lobatto-3C method (right) of order 4

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Table 2. The coefficients of Sdirk-2a (left), Sdirk-2b (middle), and Sdirk-3a (right) methods. All methods have order 2.

$\frac{2-\sqrt{2}}{2}$	$\frac{2-\sqrt{2}}{2}$	0
1	$\frac{\sqrt{2}}{2}$	$\frac{2-\sqrt{2}}{2}$
	$\frac{\sqrt{2}}{2}$	$\frac{2-\sqrt{2}}{2}$

$\frac{2+\sqrt{2}}{2}$	$\frac{2+\sqrt{2}}{2}$	0
1	$-\frac{\sqrt{2}}{2}$	$\frac{2+\sqrt{2}}{2}$
	$-\frac{\sqrt{2}}{2}$	$\frac{2+\sqrt{2}}{2}$

$\frac{3-\sqrt{3}}{6}$	$\frac{3-\sqrt{3}}{6}$	0	0
$1-\frac{\sqrt{3}}{3}$	$\frac{3-\sqrt{3}}{6}$	$\frac{3-\sqrt{3}}{6}$	0
1	$\frac{3-\sqrt{3}}{6}$	$\frac{\sqrt{3}}{3}$	$\frac{3-\sqrt{3}}{6}$
	$\frac{3-\sqrt{3}}{6}$	$\frac{\sqrt{3}}{3}$	$\frac{3-\sqrt{3}}{6}$

itself. Thus the discrete adjoint is stiffly accurate. The Butcher tableaus for these methods are presented in table 1.

Singly-diagonally-implicit Runge-Kutta methods. Sdirk-2 are 2-stage, L-stable, stiffly accurate methods of order 2. The choice $\gamma = 1 - \sqrt{2}/2$ (Sdirk-2a) is more accurate, while the choice $\gamma = 1 + \sqrt{2}/2$ (Sdirk-2b) may be advantageous when a non-negative numerical solution (concentrations) are needed.

Sdirk-3a is a 3-stage, second order, stiffly accurate method. Its coefficients are chosen such that the discrete adjoint is also stiffly accurate.

The methods Sdirk-4 are the fourth order L-stable singly-diagonally-implicit Runge-Kutta methods developed by Hairer and Wanner [14]. Specifically, Sdirk-4a is the method with $\gamma = 4/15$ and Sdirk-4b the method with $\gamma = 1/4$. The coefficients of these methods are given in [14] and not reproduced here.

Implementation Aspects. Following [14, Section IV.8], for implementation purposes (1) is written in terms of the variables $Z_i = Y_i - y^n$. Replacing the nonlinear system in k_i by a nonlinear system in Z_i has numerical advantages for stiff systems where f has a large Lipschitz constant. With the compact notation

$$Z = \Big[Z_1 \cdots Z_s \Big]^T \ , \quad F(Z) = \Big[f\,(\,T_1,\, y^n + Z_1\,) \cdots f\,(\,T_s,\, y^n + Z_s) \Big]^T \ ,$$

the nonlinear system (1) in Z is

$$Z = (hA \otimes I_n) \cdot F(Z) \ , \tag{2}$$

where $\otimes$ denotes the Kronecker product. This system can be solved by simplified Newton iterations of the form

$$\begin{array}{l} [I_{ns} - hA \otimes J\,(\,t^n,\, y^n\,)] \cdot \Delta Z^{[m]} = Z^{[m]} - (hA \otimes I)\ F^{[m]} \\ Z^{[m+1]} = Z^{[m]} - \Delta Z^{[m]} \end{array} \tag{3}$$

with J the Jacobian of the ODE function. The linear systems in (3) have dimension $ns \times ns$. For SDIRK methods (3) is a sequence of s nonlinear systems of dimension $n \times n$. For fully implicit methods a transformation of the system (3) to complex form is used following [14].

Using the implementations of the forward routines, integrators for the tangent linear model (TLM) and adjoint model (ADJ) were developed. Each family of forward methods (implicit Runge-Kutta, singly-diagonally-implicit Runge-Kutta, and Rosenbrock methods) was extended to calculate the TLM, the generated result is the sensitivity matrix.

Small perturbations of the solution (due to small changes δy^0 in the initial conditions) propagate forward in time according to the *tangent linear model*. The variation of (1) is equivalent to applying the Runge-Kutta to the sensitivity equation to get the tangent linear Runge-Kutta methods:

$$\begin{aligned} y^{n+1} &= y^n + h\sum_{i=1}^{s} b_i k_i \ , \quad \delta y^{n+1} = \delta y^n + h\sum_{i=1}^{s} b_i \ell_i \ , \\ Y_i &= y^n + h\sum_{j=1}^{s} a_{ij} k_j \ , \qquad \delta Y_i = \delta y^n + h\sum_{j=1}^{s} a_{ij} \ell_j \ , \\ k_i &= f\left(T_i,\, Y_i\right) \ , \qquad\qquad \ell_i = J\left(T_i,\, Y_i\right) \cdot \delta Y_i \ . \end{aligned} \tag{4}$$

The system (4) is linear and does not require an iterative procedure. However, even for a SDIRK method ($a_{ij} = 0$ for $i > j$ and $a_{ii} = \gamma$) each stage requires the LU factorization of a different matrix. To avoid repeated factorizations our implementation solves the linear system (4) by iterations of the form (3), re-using the LU decomposition of the matrix based on $J(t^n, y^n)$.

The adjoint methods present the final additions to the library of integrators. By solving the system in forward mode and using checkpoints to save the states at each step of the computation the sensitivity matrix is calculated in backward mode on these checkpoints.

The discrete adjoint of the Runge-Kutta method (1) is

$$\begin{aligned} u_i &= h\, J^T\left(T_i, Y_i\right) \cdot \left(b_i \lambda^{n+1} + \sum_{j=1}^{s} a_{j,i} u_j \right) \ , \quad i = s, \cdots, 1 \\ \lambda^n &= \lambda^{n+1} + \sum_{j=1}^{s} u_j \ . \end{aligned} \tag{5}$$

For $b_i \neq 0$ the RK adjoint can be rewritten as another Runge-Kutta method. To avoid repeated factorizations our implementation solves the linear system (5) by iterations of the form (3), re-using the LU decomposition of the matrix based on $J(t^n, y^n)$. The forward backward substitutions correspond to the transpose of the system (3).

4 Application and Results

In this section we illustrate the behavior of the new integrators applied to a chemical model. The computations were tested against an independent code: Fortran77 SEULEX stiff differential equations solver by Hairer and Wanner [14].

4.1 The Chemical Model

The chemical model chosen is the SAPRC99 model [18]. It includes 74 species and 211 chemical reaction equations on these species. The tests were performed on data after a 24 hour stabilizing period. The integrators were applied on a 24 hour interval. Emissions have been included in the computations.

4.2 Forward Methods

All new solvers have been tested thoroughly. The forward integrators were tested on the full range of relative error tolerances of 10^{-1} up to 10^{-8}. The absolute error tolerances were adjusted accordingly ($10^3 \cdot$ rel_tol). Results can be seen in Fig. 1. While the implicit Runge-Kutta methods Radau-2A and Lobatto-3C are still very time efficient at high accuracy, the Rosenbrock and Sdirk methods are very efficient in lower accuracy areas. The Rodas-4 integrator shows very powerful and fast performance in our application.

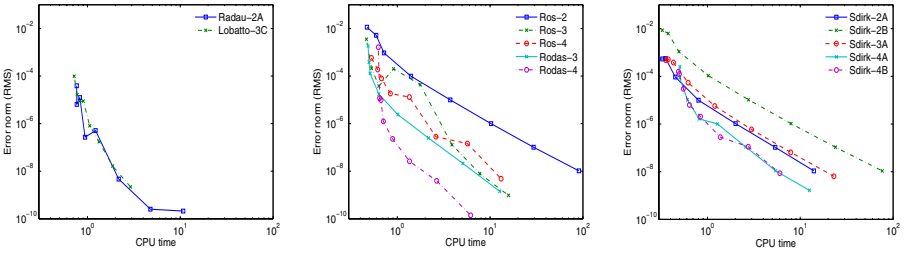

Fig. 1. Comparison of time efficiency and accuracy of forward Runge-Kutta, Rosenbrock and Sdirk methods

4.3 Tangent Linear Methods

Main feature of the new implementation are the sensitivity calculations. The Tangent Linear Model provides the background for the TLM-solvers. The tested relative error tolerances were 10^{-3}, 10^{-6}, 10^{-9}, and 10^{-12}. The default settings of Radau-2A stopped the calculations when applying the tolerance of 10^{-12} - to many steps were used, here we didn't report a result. Figure 2 presents four exemplary results when using the newly implemented integrators. Time-efficiency of Sdirk-4B is the best for any lower accuracy in our model.

4.4 Adjoint Methods

The adjoint methods also provide the sensitivity matrix. The backward scheme requires a large amount of memory storage for results of the forward computation. In order to obtain higher accuracy, even more memory will be used. We have tested our code on error tolerances of 10^{-3}, 10^{-6}, and 10^{-9}. Here, Rodas-4 outperformed Sdirk-4B. In the future we will also implement adjoints based on Radau-2A and Lobatto-3C, because they may achieve higher efficiency on increased accuracy.

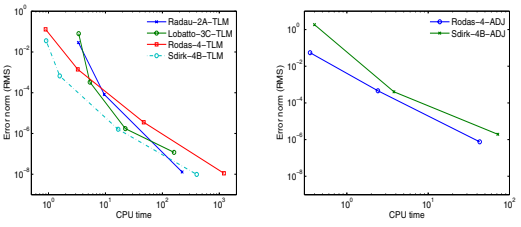

Fig. 2. Comparison of time efficiency and accuracy of tangent linear model (left) and adjoint model (right)

5 Conclusion

State-of-the-art and high-order stiff ODE integrators were added to the library of the widely-used software environment KPP. The new forward methods can achieve a high accuracy. The stability matrix of the species can be computed by using the newly implemented tangent linear model methods as well as adjoint methods. Researchers will now be enabled to easily include stability considerations into their model.

The KPP-2.2 source code is distributed under the provisions of the GNU public license [23] and is available on the web [24].

Acknowledgments

This work was supported by the National Science Foundation through the awards NSF ITR AP&IM 0205198, NSF CAREER ACI0413872, and NSF CCF0515170, by the National Oceanic and Atmospheric Administration (NOAA) and by the Texas Environmental Research Consortium (TERC).

References

1. V. Damian, A. Sandu, M. Damian, F. Potra, and G.R. Carmichael: "The Kinetic PreProcessor KPP – A Software Environment for Solving Chemical Kinetics", *Computers and Chemical Engineering*, Vol. 26, No. 11, p. 1567–1579, 2002.
2. A. Sandu, D. Daescu, and G.R. Carmichael: "Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: I – Theory and Software Tools", *Atmospheric Environment*, Vol. 37, p. 5083–5096, 2003.
3. D. Daescu, A. Sandu, and G.R. Carmichael: "Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: II – Validation and Numerical Experiments", *Atmospheric Environment*, Vol. 37, p. 5097–5114, 2003.
4. A. Sandu and R. Sander: "KPP – User's Manual", `http://www.cs.vt.edu/~asandu/Software/Kpp`.
5. A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, and F.A. Potra: " Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock methods", *Atmospheric Environment*, 31:3459–3472, 1997.

6. A. R. Curtis and W. P. Sweetenham: "Facsimile/Chekmat User's Manual", Computer Science and Systems Division, Harwell Lab., Oxfordshire, Great Britain, Aug. 1987
7. R. Djouad and B. Sportisse and N. Audiffren: "Reduction of multiphase atmospheric chemistry", *Journal of Atmospheric Chemistry*, 46:131-157, 2003.
8. R. von Glasow and R. Sander and A. Bott and P. J. Crutzen: "Modeling halogen chemistry in the marine boundary layer. 1. Cloud-free MBL",*Journal of Geophysical Research*, 107(D), EID:4341, DOI:10.1029/2001JD000942, 2002.
9. R. von Kuhlmann and M. G. Lawrence and P. J. Crutzen and P. J. Rasch: "A model for studies of tropospheric ozone and nonmethane hydrocarbons: Model description and ozone results", *Journal of Geophysical Research*, 108(D), DOI:10.1029/2002JD002893, 2003.
10. R. Sander and A. Kerkweg and P. Jöckel and J. Lelieveld: "Technical Note: The new comprehensive atmospheric chemistry module MECCA", *Atmospheric Chemistry and Physics*, 5:445-450, 2005.
11. J. Trentmann and M. O. Andreae and H.-F. Graf: "Chemical processes in a young biomass-burning plume", *Journal of Geophysical Research*, 108(D), DOI:10.1029/2003JD003732, 2003.
12. Y. Tang and G. R. Carmichael and I. Uno and J.-H. Woo and G. Kurata and B. Lefer and R. E. Shetter and H. Huang and B. E. Anderson and M. A. Avery and A. D. Clarke and D. R. Blake: "Impacts of aerosols and clouds on photolysis frequencies and photochemistry during TRACE-P: 2. Three-dimensional study using a regional chemical transport model", *Journal of Geophysical Research*, 108(D), DOI:10.1029/2002JD003100, 2003.
13. E. Hairer and G. Wanner: "Solving Ordinary Differential Equations I. Nonstiff Problems", Springer Series in Computational Mathematics, 1991.
14. E. Hairer and G. Wanner. Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer Series in Computational Mathematics, 1996.
15. J.R. Leis and M.A. Kramer: "ODESSA - An Ordinary Differential Equation Solver with Explicit Simultaneous Sensitivity Analysis", *ACM Transactions on Mathematical Software*, Vol. 14, 1:61, 1986.
16. K. Radhakrishnan and A. Hindmarsh: "Description and use of LSODE, the Livermore solver for differential equations", Lawrence Livermore National laboratory Report UCRL-ID-113855, 1993.
17. P.N. Brown and G.D. Byrne and A.C. Hindmarsh: "VODE: A Variable Step ODE Solver", *SIAM J. Sci. Stat. Comput.*, 10:1038–1051, 1989.
18. W.P.L. Carter: "Documentation of the SAPRC-99 Chemical Mechanism for VOC Reactivity Assessment", California Air Resources Board Contract, 92–329, 2000.
19. `http://pdfcentral.shriver.umbc.edu/AutoChem/`
20. `http://www.reactiondesign.com/products/open/chemkin.html`
21. `http://www.llnl.gov/CASC/odepack/`
22. `http://mcm.leeds.ac.uk/MCM/`
23. `http://www.gnu.org/copyleft/gpl.html`
24. `http://www.cs.vt.edu/$\sim$asandu/Software/Kpp`
25. `http://www.mathworks.com/products/matlab/`